ClearCase Install Guide for Linux

Table of Contents

1 Introduction

Installation of Rational® ClearCase® on Linux systems takes more consideration compared to other flavors of UNIX. There are some preparations to undertake before installing, or modifying, ClearCase on Linux (Red Hat or SuSE Enterprise).

Here is the Installation document with the advent of the vnode layer. Preparations are needed to ensure a successful installation of ClearCase on Linux.

2 Preparation of Vnode Layer for Install

IMPORTANT: ***Login as *'root'**** This is necessary for all of the following procedures. None of these steps will modify or taint the kernel. The source files are only necessary in order to successfully build the correctly configured vnode layer and subsequent module, ensuring the kernel symbols match exactly.

First and foremost, you will need to verify if the kernel source files are installed on the Linux system, and match what is currently the loaded kernel (**uname –a,** or sometimes to be 100% -> **cat /proc/version**). This can be accomplished one of two ways:

A) cd into /usr/src and run "**ls -l**". You should see something similar to the following *example* directories and links:

On Red Hat 2.4.x kernel systems

```
drwxr-xr-x    2 root    root       4096 Jan 24     2003 debug
lrwxrwxrwx    1 root    root         14 Aug 24    17:11 linux -> linux-2.4.20-8
lrwxrwxrwx    1 root    root         14 Aug 24    17:10 linux-2.4 -> linux-2.4.20-8
drwxr-xr-x   16 root    root       4096 Aug 25  12:17 linux-2.4.20-8
drwxr-xr-x    7 root    root       4096 Aug 23 19:40 redhat
```

The red is the directory with the kernel source files. (If viewing in B&W, refer to 4th line)
The blue are the *links* to the kernel source files directory. (If viewing in B&W, refer to 2nd and 3rd lines)

On SuSE 2.6.x kernel systems

```
lrwxrwxrwx   1  root   root        16    2004-11-01 16:52 linux -> linux-2.6.5-7.97
drwxr-xr-x  20  root   root      4096    2004-11-02 15:36 linux-2.6.5-7.97
drwxr-xr-x   3  root   root      4096    2004-11-01 12:02 linux-2.6.5-7.97-obj
lrwxrwxrwx  1  root   root        20    2004-11-01 16:52 linux-obj -> linux-2.6.5-7.97-obj
drwxr-xr-x  7  root   root      4096    2004-11-01 16:24 packages
```

On SuSE, the linux source directory is not the intended directory to use, but rather the linux-{kernel#}-obj directory.
The red is the directory with the kernel object source files. (If viewing in B&W, refer to 3rd line)
The blue is the *link* to the kernel object source files directory. (If viewing in B&W, refer to 4th line)

B) Or you can run the command:

"**rpm -qa | grep kernel**"

You should see something similar to this:

[root@azazel root]# rpm -qa | grep kernel
kernel-doc-2.4.21-9.EL.sgi1
kernel-source-2.4.21-4.EL
kernel-2.4.21-4.EL
kernel-utils-2.4-8.37
kernel-pcmcia-cs-3.1.31-13

You may see more output depending on how or what you installed. The things you need to see are the kernel-# and kernel-source-#.

If you don't see this, then you need to get the kernel sources installed ASAP before installing ClearCase. If the kernel source files are installed, proceed to the "*Preparing Vnode Layer*" section below.

3 Preparing Vnode Layer

This section details the necessary steps to prepare the vnode layer (and module) to be built using the kernel source files. The function of the vnode layer and module is to act as a bridge and buffer between the MVFS and the Linux kernel. This allows for the flexibility of the MVFS module, or layer, to interact with the various kernel versions that exist for Linux systems.

3.1 Red Hat/SuSE 2.4.x kernel systems

1) **cd /usr/src/**
- verify source directory is there (**ls -l**) and there is a link to either a 'linux' directory or a shortened directory such as 'linux-2.4', you should see a similar *example* output to:

drwxr-xr-x 2 root root 4096 Jan 24 2003 debug
lrwxrwxrwx 1 root root 14 Aug 24 17:11 linux -> linux-2.4.20-8
lrwxrwxrwx 1 root root 14 Aug 24 17:10 linux-2.4 -> linux-2.4.20-8
drwxr-xr-x 16 root root 4096 Aug 25 12:17 linux-2.4.20-8
drwxr-xr-x 7 root root 4096 Aug 23 19:40 redhat

If not, proceed to Step 2, otherwise skip to Step 3...

2) **ln -s *{linux-kernel-directory}* linux-2.4**
-creates link of kernel source directory to linux-2.4 directory (you can choose 'linux' instead of 'linux-2.4')
-once the sources are installed, create a link like you see above from 'linux-2.4' (or 'linux') to linux-2.4.20-8 (or to whatever your kernel source directory is). This will make things easier later on.

3) **cd linux-2.4**
-or 'linux' if that was the link name you created in step 2, or had previously.

4) **make mrproper**
-this removes any old configuration file (.config) that may be there from previously

5) Now you need to build a ".config" file to represent what is configured in your current kernel setup, and so the vnode can be built correctly with matching symbols. ****Once done with Step 5, proceed to steps 6 & 7 on page 6****

a) If you are using a Red Hat 2.4.x kernel, and can cd to /boot (may not be mounted on some systems), then:

-cd to /boot
-run '**ls –l**' to view the contents of the directory.
-look for a "config-{kernel#}" file (for example, "config-2.4.21-4.EL")
-copy this file to the top-level of the linux kernel source directory (/usr/src/linux-2.4, for example) as ".config"

Example…

cp config-2.4.21-4.EL /usr/src/linux-2.4/.config

b) If you are using a SuSE 2.4.x kernel, and can cd to /boot (may not be mounted on some systems), then:

-cd to /boot
-run '**ls –l**' to view the contents of the directory.
-look for "vmlinuz.config" file.
-copy this file to the top-level of the linux kernel source directory (/usr/src/linux-2.4, for example) as ".config"

Example…

cp vmlinuz.config /usr/src/linux-2.4/.config

c) If the mounted directory /boot is *not* available, use the following steps.

3.1a Red Hat 2.4.x

-cd into the "configs" directory (/usr/src/linux-2.4/configs, for example).

You will see, for *example*:

[root@Azazel configs]$ ls

kernel-2.4.20-athlon.config	*kernel-2.4.20-i586-smp.config*
kernel-2.4.20-athlon-smp.config	*kernel-2.4.20-i686-bigmem.config*
kernel-2.4.20-i386-BOOT.config	*kernel-2.4.20-i686.config*
kernel-2.4.20-i386.config	*kernel-2.4.20-i686-smp.config*
kernel-2.4.20-i386-smp.config	*kernel-2.4.20-x86_64.config*
kernel-2.4.20-i586.config	*kernel-2.4.20-x86_64-smp.config*

-choose the config file that best represents your server hardware setup and copy it up one directory as ".config" (running the command "uname –a" can help determine this)

Example...

cp kernel-2.4.20-i686.config ../.config
cd ..
ls –al (to confirm '.config' is there)

3.1b SuSE 2.4.x (SLES 9.0)

-cd into the "arch" directory (/usr/src/linux-2.4/arch, for example).

You will see for *example*:

```
[root@azazel /usr/src/linux/arch] # ls -al
total 5
drwxr-xr-x  20 root  root   480 Jan  9  2004 .
drwxr-xr-x  17 root  root   744 Nov 20 01:24 ..
drwxr-xr-x   7 root  root   288 Jan  9  2004 alpha
drwxr-xr-x  18 root  root   680 Jan  9  2004 arm
drwxr-xr-x   7 root  root   312 Jan  9  2004 cris
drwxr-xr-x   8 root  root   312 Jan  9  2004 i386
drwxr-xr-x  12 root  root   408 Jan  9  2004 ia64
drwxr-xr-x  20 root  root   632 Jan  9  2004 m68k
drwxr-xr-x  29 root  root  1560 Jan  9  2004 mips
drwxr-xr-x  11 root  root   624 Jan  9  2004 mips64
drwxr-xr-x   7 root  root   288 Jan  9  2004 parisc
drwxr-xr-x  12 root  root   408 Jan  9  2004 ppc
drwxr-xr-x   9 root  root   336 Jan  9  2004 ppc64
drwxr-xr-x   7 root  root   328 Jan  9  2004 s390
drwxr-xr-x   6 root  root   304 Jan  9  2004 s390x
drwxr-xr-x   7 root  root   288 Jan  9  2004 sh
drwxr-xr-x   8 root  root   312 Jan  9  2004 sparc
drwxr-xr-x   9 root  root   336 Jan  9  2004 sparc64
drwxr-xr-x  13 root  root   752 Jan  9  2004 um
drwxr-xr-x   9 root  root   336 Jan  9  2004 x86_64
```

-choose the directory that matches the architectuire you are using, then cd into it (i.e. 'cd i386'):

```
[root@azazel /usr/src/linux/arch]# cd i386
[root@azazel /usr/src/linux/arch/i386] # ls -al
total 76
drwxr-xr-x   8 root  root   312 Jan  9  2004 .
drwxr-xr-x  20 root  root   480 Jan  9  2004 ..
-rw-r--r--   1 root  root  3797 Oct 21  2002 Makefile
drwxr-xr-x   4 root  root   240 Jan  9  2004 boot
-rw-r--r--   1 root  root 15363 Oct 21  2002 config.in
-rw-r--r--   1 root  root 46245 Oct 21  2002 defconfig
drwxr-xr-x   2 root  root   352 Jan  9  2004 kdb
drwxr-xr-x   2 root  root  1496 Nov 20 01:28 kernel
drwxr-xr-x   2 root  root   384 Nov 20 01:28 lib
drwxr-xr-x   2 root  root  1520 Jan  9  2004 math-emu
drwxr-xr-x   2 root  root   240 Nov 20 01:28 mm
-rw-r--r--   1 root  root  1909 Oct 21  2002 vmlinux.lds.S
```

-copy the defconfig file to the top-level kernel source directory as .config

```
cp defconfig ../../.config
cd ../..
ls –al     (to confirm '.config' is there)
```

6) **make oldconfig** (run in top-level of kernel source directory, i.e. /usr/src/linux-2.4)
-this strips down the original config file to what is actually configured for your system.

7) **make dep** (run in top-level of kernel source directory, i.e. /usr/src/linux-2.4)
- this makes the dependencies
- This will take a bit as it checks for and resolves any dependencies based on kernel settings.

3.2 On SuSE 2.6.x kernel systems (SLES 9.1+)

1) **cd /usr/src/**
- verify kernel object source directory is there (**ls -l**) and there is a link to either a 'linux' directory or a shortened directory such as 'linux-2.4', you should see a similar *example* output to:

lrwxrwxrwx 1 root root 16 2004-11-01 16:52 linux -> linux-2.6.5-7.97
drwxr-xr-x 20 root root 4096 2004-11-02 15:36 linux-2.6.5-7.97
drwxr-xr-x 3 root root 4096 2004-11-01 12:02 linux-2.6.5-7.97-obj
lrwxrwxrwx 1 root root 20 2004-11-01 16:52 linux-obj -> linux-2.6.5-7.97-obj
drwxr-xr-x 7 root root 4096 2004-11-01 16:24 packages

If not, proceed to Step 2, otherwise skip to Step 3...

2) **ln -s *{linux-kernel-object-directory}* linux-obj**
-creates link of kernel object source directory to linux-obj directory.
-once the sources are installed, create a link like you see above from 'linux-obj' to *linux-2.6.5-7.97-obj* (or to whatever your kernel object source directory is). This will make things easier later on.

3) **cd linux-obj**
-or whatever you used to create a link to your kernel object source directory
- If you cd into that link you will see, for *example* (could be i686 if that is :

root@azazel:/usr/src> cd linux-obj/
root@azazel:/usr/src/linux-obj> ll

total 4
drwxr-xr-x 6 root root 4096 2004-11-01 12:02 i386

Then, cd into the 'i386' directory. You will something similar to the following *example*:

root@azazel:/usr/src/linux-obj# cd i386/
root@azazel:/usr/src/linux-obj/i386# ll
total 16
drwxr-xr-x 6 root root 4096 2004-11-01 12:02 bigsmp
drwxr-xr-x 6 root root 4096 2004-11-01 12:02 debug
drwxr-xr-x 6 root root 4096 2004-11-04 18:38 default
drwxr-xr-x 6 root root 4096 2004-11-01 12:02 smp

Here, you would choose either the 'smp' or 'default' directories, depending on your current configuration (from kernel# of uname –a output). Next, cd into the directory you chose. For this *example*, we will assume a default configuration:

root@azazel: /usr/src/linux-obj/i386# cd default
root@azazel:/usr/src/linux-obj/i386/default> ls -al
total 156
drwxr-xr-x 6 root root 4096 2004-11-04 18:38 .
drwxr-xr-x 6 root root 4096 2004-11-01 12:02 ..
drwxr-xr-x 3 root root 4096 2004-11-01 12:02 arch
-rw-r--r-- 1 root root 55534 2004-11-04 18:38 ***.config***
-rw-r--r-- 1 root root 4886 2004-11-04 18:38 .config.cmd
-rw-r--r-- 1 root root 55534 2004-07-02 10:45 .config.old
drwxr-xr-x 4 root root 4096 2004-11-01 12:02 include
drwxr-xr-x 2 root root 4096 2004-11-01 12:02 include2
-rw-r--r-- 1 root root   321 2004-07-02 10:45 Makefile
drwxr-xr-x 5 root root 4096 2004-11-01 12:02 scripts

So, the path to this .config file is "/usr/src/linux-obj/i386/default". There is NO need to run the usual "make oldconfig" and "make dep" commands, as this .config file is already preconfigured.

3.3 On SuSE 2.4.x kernel systems (SLES 8)

Verify that you have the following directory:

/usr/src/linux-include/default

or…

/usr/src/linux-{kernel#}-include/default

If not, make sure the appropriate kernel source files are installed, and this directory structure will be created. If this exists, then we will use this path during installation when prompted to link the MVFS. There is already the proper configured ".config" file in this path for the currently loaded kernel. During installation of Clearcase, input '*yes*' to link the MVFS, then input the correct path to your kernel include path from above.

3.4 On Red Hat Enterprise 4 2.6.x kernel systems (RHEL4)

The new operating system by Red Hat is based off of the 2.6.x kernel. As of this writing (5/26/05), IBM Rational Clearcase only supports the base version of RHEL4 (NOT update 1) due to known module compilation issues. Support is only offered with patches 42 (user space) and 43 (MVFS) for version 2002.05, and with patches 20 (MVFS) and 21 (User Space) for version 2003.06. There are two steps to consider for installation of clearcase:

1) [Recommended] During installation of clearcase, when asked to relink the MVFS, say "NO". This will tell clearcase to load the precompiled modules that ship with the SR5 patches. These should work fine. If this fails, try step 2…
2) During installation, choose to relink the MVFS, and when asked for the kernel source directory, input the following…

   /lib/modules/*{uname –r}*/build

   This is a link to the correct kernel source directory path.

****

You are ready to Install ClearCase, refer to Section 4 *Installation of Clearcase*. When asked to rebuild or link the vnode you will use "/usr/src/linux-obj/i386/default" (or whatever applies to your environment for the selected path) to point to the correct .config file. If you are still having any problems installing ClearCase, please call IBM Rational Support.

4 Installation of ClearCase

Installing when there is no patching required

Download the install bits or create the release area from the cd-rom. You can create any directory structure anywhere you like.

You should have already contacted IBM Rational ClearCase Licensing and received your ClearCase licenses. You should, also, have already determined what server(s) will be the VOB, view, license and registry servers. Once you have done that, simply cd down the release area, usually /<dir>/2003.06/<platform like sun5>/install and run "**./site_prep**" and fill in the appropriate values for the questions that will populate the site.dat file used by "install_release" script.

Then, run "**./install_release**" and select your install method and modes (local and Full installation is recommended). You have already decided which systems are going to be the license and registry servers (during the site_prep). For a VOB server install, you will need to have determined ahead of time whether or not you are going to use schema 54 for large VOB support or not (schema 53). The schema version only deals with the size of the database files in the /db directory and NOT on the actual file elements, source code.

At some point you will be asked if you want to relink the vnode to the kernel, here you will answer *yes*, and when prompted for the path, make sure the absolute path to the kernel source directory is there (enter it if not). From the above example, you would enter "/usr/src/linux-2.4". Finish up the installation.

If you receive errors, then you may need some help from IBM Rational Support. If you receive 1 or more lines about unresolved symbols, please refer to the section below entitled "*Rebuilding the MVFS and VNODE Modules*". If you do not get those errors or only get warnings, then you can test your installation by trying some cleartool commands, once you get the license and registry servers up and running. You should be all set then with no errors attempting the cleartool commands. To verify if the mvfs module loaded successfully perform a "**cleartool –ver**" command and look for an output with an MVFS version like the following *example*:

ClearCase version 2003.06.00 (Tue Dec 16 21:15:58 EST 2003)
clearcase_p2003.06.00-4 (Fri Apr 02 18:06:11 EST 2004)
clearcase_p2003.06.00-5 (Fri Apr 02 17:32:59 EST 2004)
clearcase_p2003.06.00-7 (Tue Jun 01 11:07:22 EDT 2004)
clearcase_p2003.06.00-8 (Wed Jun 02 11:57:26 EDT 2004)
@(#) MVFS version 2003.06.10 (Sat Apr 10 22:06:07 EDT 2004) VNODE built $Date: 2004-12-16.18:10:24 (UTC) $
cleartool 2003.06.10+ (Sat Apr 10 21:48:18 EDT 2004)
db_server 2003.06.10+ (Sat Apr 10 21:43:47 EDT 2004)
VOB database schema version: 54

Repeat this process for the view server and registry server. For the license server, you will need to install ClearCase then stop it. Create a license.db file in the /var/adm/atria/ directory, copy the license string info from Rational Licensing, and restart ClearCase for it to take affect.

4.1 Upgrading the Kernel

If you upgrade the Linux kernel from an officially sanctioned default kernel from RedHat or SuSE, make sure to get the kernel sources as well. If you recompile the kernel to take advantage of some new feature for hardware, software, etc, then you will need to replay the preparation steps discussed earlier to creating a new ".config" and reinstall per Patchless Installation. Please refer to the section below entitled "*Rebuilding the MVFS and VNODE Modules*".

4.2 Patching the Release Area

There will be a time when you will need to apply ClearCase patches to the release area.

-cd to /<dir>/2003.06/<platform>
-Create a directory there called 'patches'
-From here on in, all patch tarballs will go here for installation.

Let's say you want to apply patch 1, for *example*:

Download and transfer the file ClearCase_p2003.06-1.tar.gz to the patches directory. Then, gunzip and untar the file. This will create a directory called clearcase_p2003.06-1. Cd down to the install directory. Find the apply_patch script file and run "**./apply_patch**". It will output files it is changing to the release area, applying the patch files. Once it is finished, you will need to reinstall ClearCase per the Installation section.

Hint: If you are not the original installer of ClearCase or it has been sometime since you did, I suggest if you are in a Gnome/KDE environment, opening another terminal session and opening the last Rational install log found in /var/adm/atria/log. Look for the latest timestamp on the name of the file. This will have the last settings used for that install.

4.3 Summary of Install Steps

"**./install_release**" (or ./site_prep if you haven't run already to create a site.dat file which stores all the info needed for the install)

- choose 'Local Install'
- press x to bypass agreement, and enter yes to question
- choose 'Full Install'
- select 'Full Clearcase' and/or 'Full Multisite' (as well as any of the other choices that are applicable, such as Rational Web Platform – RWP)
- select defaults previously inputted for registry/license server etc.
- select "*yes*" to link Vnode layer to MVFS
- enter path to kernel source directory from steps 2 & 3 (/usr/src/linux-2.4, in *example*)
- answer "*yes*" to extended VOB support if you want Schema 54, "*no*" for schema 53

5 Rebuilding the MVFS and VNODE Modules

This section is for manually rebuilding the vnode module/layer when the Preparation and Installation steps fail, or when you make a modification to the currently loaded kernel and have already installed ClearCase previously. If you have followed the Preparation and Installation steps, and the MVFS still fails to load and/or you get unresolved symbols, refer to the following steps.

If you are running a Linux kernel that is compatible with one of the pre-built vnode modules, you should not need to rebuild the module. Use the binary versions shipped in this release instead.

The kernel configuration files used to build the binary vnode modules can be found in *.config files in /opt/rational/clearcase/etc/conf/linux* location. The file names take the form kernel-<version>-<platform>[-smp].config, where

<version> is the Linux kernel version.
<platform> is the hardware platform. (i586, s390)

If you try to load a vnode module built for a matching kernel revision, but see errors like the following, you need to rebuild your vnode module:

/lib/modules/fs/vnode.o: unresolved symbol d_rehash_0680a5aa
/lib/modules/fs/vnode.o: unresolved symbol d_alloc_054621f7
/lib/modules/fs/vnode.o: unresolved symbol __mntput_def2fc0a
/lib/modules/fs/vnode.o: unresolved symbol inode_setattr_e5f5e28a

6 Relink Instructions for ClearCase 2003.06.00

After you install ClearCase, boot the kernel you want to use, and become the super-user. Then build the vnode module as follows:

1) Stop ClearCase

/opt/rational/clearcase/etc/clearcase stop (or… **/usr/atria/etc/atria_start stop**)

2) If you are using SuSE 2.6.x kernel system, you can skip to Step 5, otherwise verify that neither the mvfs.o nor vnode.o files exist in the directory /lib/modules/`uname -r`/fs. If you find either file, delete or rename it. If these object files remain from a previous version of ClearCase, they will interfere with loading of the new version.
Warning: Do NOT remove /lib/modules/fs/mvfs.o or /lib/modules/fs/vnode.o.

3) Verify that your top level Linux kernel source directory contains a ".config" file that matches your actual system configuration. If you are running a kernel that you have patched, there should be one from the last time that you rebuilt your kernel.

If you are using SuSE 2.6.x kernel system, you can skip to Step 5.

If you are running a system with a default kernel from your distribution, look for a ".config" file that matches your system, and use it to configure a kernel source tree.

* For Red Hat, look in the configs subdirectory of the kernel sources.
* For SuSE on x86, look in /boot/vmlinuz.config.
* For SuSE on 390, look in /boot/kernel/.config
* For other distributions, check your vendor documentation to help locate the config file matching your installed kernel.

Copy the matching configuration file to ".config" in the top level of your kernel source directory.

3a) If you are running a custom configured kernel and you can't find a matching .config file included with your distribution, you must run make xconfig or make menuconfig to regenerate your ".config"

file. After you have built a configuration file, run make dep to build dependencies. Then proceed to step 5.

4) If necessary, build or rebuild the generated headers in the Linux source tree:

cd <*linux-kernel-source-directory*>
make oldconfig
make dep

5) Change directory to the MVFS vnode build directory:

cd /var/adm/rational/clearcase/mvfs/vnode_src

6) Build and install the vnode module. The build process queries you for the directory containing your Linux kernel sources (LINUX_KERNEL_DIR), and attempts to discern the kernel patch level by examining the output of /bin/uname and the contents of release information files in /etc/SuSE-release or /etc/redhat-release. The script stores the determined settings in a configuration file.

6a) Clean up from previous builds:

make clean

6b) Set options for the current running kernel, choose the linux source directory:

make vnode_param.mk.config

6c) If the previous step fails, create the ".config" file manually.
Set the following two values in the file:

RATL_EXTRAFLAGS=<appropriate value>
LINUX_KERNEL_DIR=<root of kernel source tree>

RATL_EXTRAFLAGS is needed to distinguish between certain variants of kernel sources that cannot be differentiated based solely on the 3-part kernel version (for example, 2.4.18).

To see which flags might be needed for your kernel, look at the RATL_EXTRAFLAGS array in /opt/rational/clearcase/install/Kernel.pl. Set your RATL_EXTRAFLAGS variable appropriately.

6d) If you built your kernel with a compiler other than the "cc" defined in your search path, you must specify the same compiler to use for building the vnode module. Add a line to vnode_param.mk.config specifying the compiler to be used. The following example specifies the gcc compiler:

CC=/usr/local/bin/gcc

6e) Build and install the module:

make
make install

You should not see any errors like the following:

/usr/include/linux/string.h:8:2: #warning Using kernel header in userland!

/usr/include/linux/autoconf.h:1:2: #error Invalid kernel header included in userspace

Such errors mean that the LINUX_KERNEL_DIR argument you specified was not a properly configured kernel source directory. You must use a configured kernel source tree to relink the vnode module.

TROUBLESHOOTING NOTE

If the vnode module compilation fails with errors about undefined Functions, mismatch kernel, different GCC compiler versions, or structure members, the script may have guessed the wrong kernel variant. Run your text editor on "Makefile" in your top level kernel source directory (i.e. /usr/src/linux-2.4). Look for code that is conditional on RedHat<number> or RATL_EXTRAVERSION, and make sure it is equal to your kernel variant. It may read some kernel extension like '-9.ELcustom' when you may be using the kernel '2.4.21-9.ELsmp', so you would need to edit the value for this flag to read: '-9.ELsmp'. Save the "Makefile", and run the steps in "*Preparing Vnode Layer*", then reinstall. The MVFS and vnode modules should load. Or, make the modifications to the vnode_param.mk.config file in /var/adm/atria/mvfs/vnode_src directory to set these flags in: RATL_EXTRAFLAGS, and rebuild the module manually.

Also, it is critical that you use the same gcc compiler version (gcc –v) to build the vnode module that was used to build your linux kernel ('cat /proc/version' will show what gcc compiler was used). Both outputs should match. Refer to step 6d for solution, or install the correct compiler version.

make cleano
make

7) Restart ClearCase:

/opt/rational/clearcase/etc/clearcase start

If all goes well, you should see a line in a '**cleartool –ver**' output referencing "MVFS version" and "vnode built on <*date*>" . To make sure the appropriate modules did in fact load, run '**lsmod**' to list the currently loaded modules, and look for sunrpc, mvfs, and vnode in the output.

6.1 Possible Errors

If the vnode module builds and installs cleanly but you get error messages on your console about structure size mismatches when the mvfs module is loaded, your vnode layer may include changes (either made by you or as a result of operations in this procedure) that are incompatible with MVFS.

MVFS attempts to identify these cases by checking the sizes of its compiled-in data structures with the sizes of structures used to build the vnode module. If there are any mismatches, MVFS will print them out to the console when you attempt to load the module, and the module load will fail.

To correct this, you must build a vnode module from unmodified

sources. The sources were originally installed in /var/adm/rational/clearcase/mvfs/vnode_src, and an original copy can be found in /opt/rational/clearcase/etc/conf/vnode_src.

If you can load a pre-built vnode module shipped with ClearCase, or one built from unmodified sources, but cannot load the companion MVFS module, please contact Customer Support.

6.2 Possible Warnings

If you get warnings when loading mvfs.o about mismatched kernel versions or about loading GCC-2 compiled modules in a GCC-3 kernel, you can safely ignore them if the module loads successfully.

If you get a warning when loading vnode.o about mismatched kernel versions, you can safely ignore it if the module loads successfully.
If you get a warning when loading vnode.o about loading GCC-2 compiled modules in a GCC-3 kernel, you must rebuild the vnode module with a compiler version that matches your kernel's compiler.

7 Credits

Author: Keith Lefrancois (Rational Clearcase TSE and Linux SME)
Special Thanks: Dave Hyer, Kevin Cool, William Penny, and Rich Rakich